An Improved Separation of Regular Resolution from Pool Resolution and Clause Learning (Extended Abstract)[1]

Maria Luisa Bonet[*]
Lenguajes y Sistemas Informáticos
Universidad Politécnica de Cataluña
Barcelona, Spain
bonet@lsi.upc.edu

Sam Buss[†]
Deptartment of Mathematics
University of California, San Diego
La Jolla, California, USA
sbuss@math.ucsd.edu

Abstract

We establish the unexpected power of conflict driven clause learning (CDCL) proof search by proving that the sets of unsatisfiable clauses obtained from the guarded graph tautology principles of Alekhnovich, Johannsen, Pitassi and Urquhart have polynomial size pool resolution refutations that use only input lemmas as learned clauses. We further show that, under the correct heuristic choices, these refutations can be carried out in polynomial time by CDCL proof search without restarts, even when restricted to greedy, unit-propagating search. The guarded graph tautologies had been conjectured to separate CDCL without restarts from resolution; our results refute this conjecture.

1 Introduction

Modern-day conflict-driven clause learning (CDCL) algorithms have proved to be remarkably successful in solving the satisfiability problem for applications that arise in many practical areas. These CDCL algorithms use a version of the Davis-Putnam-Logemann-Loveland methods, augmented with clause learning. Surprisingly, CDCL can solve problems of genuine interest that involve tens of thousands, or even hundreds of thousands, of propositional variables. It is known that CDCL search algorithms that use frequent restarts can simulate full resolution ([Pipatsrisawat and Darwiche, 2011]). However, it is an open question whether CDCL without restarts also simulate full resolution.

[Alekhnovich *et al.*, 2007] solved the long-standing open problem of whether regular resolution polynomially simulates resolution by giving two examples of tautologies that yield unsatisfiable set of clauses which require regular resolution refutations which are nearly exponentially longer than the shortest resolution refutations. A third such example was given by [Urquhart, 2011]. These three principles were generally conjectured to separate CDCL without restarts from resolution.

A series of recent works has refuted these conjectures. The present paper describes the first of these results, and full details can be found in [Bonet and Buss, 2012a] and [Bonet and Buss, 2012b], as well as in the subsequent work [Bonet *et al.*, 2012]. These papers prove that the guarded graph tautologies, GGT_n, of [Alekhnovich *et al.*, 2007] have polynomial size refutations in CDCL without restarts, provided the CDCL algorithm makes the right heuristic choices for decision literals and for clause learning. These results hold for a CDCL algorithm which learns clauses based on unit propagation and is required to greedily process conflicts. Greedily processing conflicts means that when the algorithm finds a conflict, it immediately produces a learned clause and backtracks without continuing the search. It is important to consider greedy algorithms, since all practical implemented algorithms are in fact greedy. In contrast, logical proofs, such as in resolution, permit non-greediness.

Our proof methods do rely heavily on formalized logical proofs however. Our upper bounds for the size of an optimal refutation generated by CDCL without restarts are based on a formalization of the algorithmic CDCL proof search in terms of a resolution proof system called "regWRTI" for "tree-like regular w-resolution with input lemmas". (In fact, we give refutations in a more restrictive system called "regRTI".) The proof system regWRTI is known to faithfully simulate the power of (non-greedy) CDCL algorithms without restarts assuming the CDCL algorithm makes the optimal heuristic choices for decision literals and for clauses to learn ([Buss *et al.*, 2008]). The theorems reported below in Section 4 below are proved by first proving the existence of the appropriate regWRTI refutations, and then giving additional arguments that convert these regWRTI refutations into *greedy* proof searches based on CDCL without restarts.

In more recent work, [Bonet *et al.*, 2012] and [Buss and Kołodziejczyk, 2012] have shown that the other two tautologies considered by [Alekhnovich *et al.*, 2007] and [Urquhart, 2011] also have polynomial size regWRTI refutations. Because of the complexity of the arguments involving greedi-

[*]Supported in part by grant TIN2010-20967-C04-02.

[†]Supported in part by NSF grant DMS-1101228.

[1]The paper on which this extended abstract is based was the recipient of the best paper award of the 2012 International Conference on Theory and Applications of Satisfiability Testing [Bonet and Buss, 2012a]. The results were reported also in the preprint [Bonet and Buss, 2012b]. The in-preparation full version that paper, [Bonet *et al.*, 2012], is joint with J. Johannsen and contains additional results.

ness, it has not been proved that these CDCL proof searches can be made greedy. It is nonetheless strongly conjectured that these CDCL algorithms can indeed be made greedy. Therefore, at the present state of knowledge, there are no plausible conjectures of tautologies for separating the refutations which can be generated by CDCL algorithms without restarts from full resolution. On the other hand, attempts to simulate full resolution by CDCL algorithms without restarts and without adding additional variables (for the latter, see [Beame *et al.*, 2004; Van Gelder, 2005; Hertel *et al.*, 2008; Buss *et al.*, 2008]) have so far been unsuccessful.

The next section describes the background and history of the work in more detail. Section 3 provides definitions. The concluding section states our main theorems.

2 Background

The problem of determining the satisfiability of a set of clauses is one of the central algorithmic problems for both computational complexity and automated theorem proving. Resolution-based proof search algorithms are of crucial importance for a wide range of automated proof search algorithms. The most broadly successful modern-day resolution proof search algorithms are based on the Davis-Putnam procedure ([Davis and Putnam, 1960]) and the improved search method of [Davis *et al.*, 1962]. These algorithms, called "DPLL proof search" or "DLL proof search", have undergone extensive development since the early 1960's.

The development of *conflict driven clause learning* (CDCL) by [Marques-Silva and Sakallah, 1999], and its implementation in the GRASP system, yielded a major improvement in the performance of DPLL proof search algorithms. The intuitive idea of CDCL is that, given a set Γ of clauses, a trial assignment of boolean values is made to variables until unit propagation reveals that the set Γ is unsatisfiable. This unsatisfiability is called a "conflict". From the conflict, a new clause is derived using input resolution (sometimes also called "trivial resolution"). The new clause is then "learned", namely added to the set Γ of clauses. At the same time, some prior learned clauses may be discarded from Γ (that is, forgotten) by garbage collection. The search algorithm now backtracks in the assignment of values to variables until there is no longer a conflict from unit propagation, and then resumes assigning boolean values to variables. This process continues until obtaining either an unconditional conflict (the empty clause) or a satisfying assignment. There are many refinements to CDCL that can give large performance improvements, including restarts, improved clause learning strategies, fast backtracking, unlearning clauses, refined heuristics for selecting boolean assignments to variables, lookahead, parallelization, lazy data structures, as well as many others.

All known CDCL proof search algorithms can be polynomially simulated by resolution. Namely, if a CDCL algorithm runs for n time steps and finds that Γ is unsatisfiable, then there is a resolution refutation of Γ in which the number of resolution inferences is bounded polynomially (even, linearly) by n. We are interested in the converse question of whether CDCL proof search without restarts can polynomially simulate resolution. CDCL proof search algorithms have some flexibility in how they choose boolean assignments to variables (the choice of "decision literals") and in how they choose to learn clauses, and in how they choose to forget learned clauses with garbage collection. In practical systems, heuristic methods are used to make these choices. Informally, we say that a CDCL proof search algorithm without restarts can polynomially simulate resolution provided that, given a set Γ of clauses which has a resolution refutation with n inferences, there exist choices for the decision literals and for learned clauses and clauses that are discarded by garbage collection that allow the CDCL algorithm to establish the unsatisfiability of Γ in time polynomially bounded by n without the need of restarts.

It is already known that frequent use of restarts allows CDCL algorithms to simulate resolution. The first such result was by [Beame *et al.*, 2004] who showed that *non-greedy* CDCL proof search *with restarts* can polynomially simulate resolution refutations. A CDCL proof search algorithm is *greedy* if it never ignores conflicts. Conversely a *non-greedy* CDCL proof search is allowed to selectively ignore conflicts that arise from unit propagation and continue assigning boolean values to variables. The advantage of non-greedy search is that it potentially allows learning clauses that could not otherwise be learned. That is to say, the CDCL algorithm might wish to learn a particular clause, but have to ignore already obtained conflicts in order to generate an alternative conflict which learns the desired clause. A non-greedy search algorithm would be allowed to do this, but a greedy search could not. The disadvantages of non-greedy search are that it is unnatural and unrealistic, and that it is not used in any implemented proof search algorithm. Thus it is undesirable to consider non-greedy CDCL proof search.

The second such result showing a simulation of resolution by clause learning was by [Pipatsrisawat and Darwiche, 2011] building on work of [Atserias *et al.*, 2011]. They avoided the non-greedy property, and proved that greedy CDCL proof search with restarts can polynomially simulate resolution.

Restarts are well-known to be an important strategy that improves the performance of CDCL proof search. Nonetheless, it is a much more interesting question whether CDCL without restarts can polynomially simulate resolution. There are several reasons for this, but primarily it is because the CDCL proof search algorithm is most elegantly viewed as being a DPLL proof search augmented with clause learning. Without restarts, the CDCL proof search algorithm is *complete* in the sense that it is guaranteed to eventually terminate, either by learning the empty clause or by finding a satisfying assignment. In contrast, the construction of [Atserias *et al.*, 2011] and [Pipatsrisawat and Darwiche, 2011] uses extremely frequent restarts; in fact, their construction allows (even prefers) that a restart occur after each clause is learned. With such frequent restarts, CDCL is no longer a "proof search" algorithm in the usual sense of the terminology. This use of frequent restarts is also at odds with the idea that one of the purposes of restarts is to avoid a "heavy tail" of poor heuristic choices for CDCL search ([Gomes *et al.*, 2000])

Practical implementations of CDCL algorithms tend to work best with a mix of quick restarts and longer DPLL proof searches. Hence it is certainly of interest to understand the power of CDCL without restarts.

As discussed in the introduction, [Alekhnovich *et al.*, 2007] and [Urquhart, 2011] gave three examples of unsatisfiable set of clauses that require exponentially longer regular resolution refutations than resolution refutations. It is well-known that non-greedy CDCL proof search without restarts can simulate regular resolution, assuming the CDCL search makes the right choices for decision literals and learned clauses. Therefore, a superpolynomial separation of CDCL without restarts from resolution also implies a superpolynomial separation between regular resolution and resolution. Since there was no obvious way for CDCL without restarts to establish the unsatisfiability of these three sets of clauses, it was conjectured that they also provide a near-exponential separation of CDCL without restarts from resolution. The first of these conjectures was refuted by [Bonet and Buss, 2012a] and [Bonet and Buss, 2012b], as we report in the present paper. The other two conjectures are refuted by the forthcoming work of [Bonet *et al.*, 2012] and [Buss and Kołodziejczyk, 2012].

Our results also provide a natural combinatorial principle that gives an exponential separation between regular resolution and CDCL proof search without restarts. An earlier exponential separation due [Van Gelder, 2005] has the disadvantage of using proof-trace variables.

The next section gives definitions, and more background material. In particular, we describe the proof system regWRTI which is closely related to the power of CDCL without restarts. As a proof system, regWRTI is inherently non-greedy. This is because proofs are inherently non-algorithmic, and even when an easy proof is at hand, there is no prohibition against using an alternate more complicated proof that helps with clause learning. For our proofs, it is helpful to first give polynomial size regWRTI refutations; by [Buss *et al.*, 2008], these regWRTI refutations automatically correspond to non-greedy CDCL without restarts. We then have to prove an additional theorem showing that the constructed regWRTI refutations actually yield polynomially long searches with *greedy* CDCL without restarts.

3 Definitions

Propositional variables range over the values *True* and *False*. A *literal* is either a variable x or a negated variable $\overline{x}$. A *clause* C is a set of literals, interpreted as the disjunction ($\vee$) of its members.

Definition 1 *A resolution inference has two premise clauses A and B and a resolution literal x, and produces a new clause C called the* resolvent.

$$\frac{A \qquad B}{C}$$

It is required that $\overline{x} \notin A$ and $x \notin B$. The different forms of resolution are:

Resolution rule. *Here $A := A' \vee x$ and $B := B' \vee \overline{x}$, and C is $A' \vee B'$.*

Degenerate resolution rule. [Hertel *et al.*, 2008; Van Gelder, 2005] *If $x \in A$ and $\overline{x} \in B$, we apply the resolution rule to obtain C. If $x \in A$ and $\overline{x} \notin B$, then the resolvent C is B. If $x \notin A$ and $\overline{x} \in B$, then the resolvent C is A. If neither A nor B contains the literal x or $\overline{x}$, then C is the lesser of A or B according to some tiebreaking ordering of clauses.*

w-resolution rule. [Buss *et al.*, 2008] *Here C is the clause $(A\setminus\{x\})\vee(B\setminus\{\overline{x}\})$. If the literal $x \notin A$ (resp., $\overline{x} \notin B$), then it is called a* phantom literal *of A (resp., B).*

The "w-resolution" inference combines a weakening inference and a resolution inference: "w" stands for "weak".

For Γ a set of clauses, a *resolution refutation* of Γ is a derivation of the empty clause from Γ using resolution. The notions of *degenerate resolution refutation* and *w-resolution refutation* are defined similarly. All three forms of resolution are sound and complete.

A refutation is *tree-like* if its underlying graph is a tree. A resolution derivation is *regular* provided that, along any path in the directed acyclic graph, each variable is resolved at most once. For degenerate and w- resolution inferences, the resolution variable may be a phantom and not appear in any clause in the inference, but this still counts as an occurrence of the variable for purposes of the definition of a regular refutation.

We define "pool resolution" using the conventions of [Buss *et al.*, 2008], who called this concept "tree-like regular resolution with lemmas". The original definitions of [Van Gelder, 2005] and [Hertel *et al.*, 2008] were slightly less restrictive. The idea is that any clause appearing in the refutation is a learned "lemma" and can be used freely from then on.

Definition 2 *The* postorder *ordering $<_T$ of the nodes in a tree T is defined so that if u is a node of T, v is a node in the subtree rooted at the left child of u, and w is a node in the subtree rooted at the right child of u, then $v <_T w <_T u$.*

Definition 3 *A* pool resolution *refutation of a set Γ of clauses is a tree T of clauses that fulfills the following conditions: (a) each leaf is labeled either with a clause of Γ or with a clause (called a "lemma") that appears earlier in the tree under the postorder $<_T$; (b) each internal node is labeled with a clause and a literal, and the clause is obtained by resolution from the clauses labeling the node's children, by resolving on the given literal; (c) the proof tree is regular; (d) the root is labeled with the empty clause.*

The notions of *degenerate pool resolution* refutation and *pool w-resolution* refutation are defined similarly. Degenerate or w-resolution may be advantageous over resolution for pool refutations by permitting more lemmas to be derived. A pool w-resolution refutation is the same as a "regWRTL" refutation in the terminology of [Buss *et al.*, 2008]. A pool resolution refutation is called a "regRTL" refutation.

Pool refutations are an alternate method of representing dag-like refutations. A lemma corresponds to a reuse of an earlier-derived clause. The advantage of expressing a refutation as a pool refutation instead of as a dag-like refutation is that it let us define "regularity" in terms of the tree structure of the pool refutation instead of talking about regular depth-first traversals of a dag. It also lets us define the notion of "input lemma":

Definition 4 *A "lemma" in part (a) of Definition 3 is called an* input lemma *if it is derived by an* input *subderivation, namely by a subderivation in which each inference has at least one hypothesis which is a member of* Γ *or a lemma. A* pool w-resolution refutation with input lemmas (regWRTI) *is a pool w-resolution refutation in which all lemmas are input lemmas.*

The motivation for considering input lemmas comes from [Beame *et al.*, 2004], who showed that the clauses learned during CDCL can be characterized in terms of clauses that can be inferred by input resolution. In large part, this is because input resolution is known to have the same logical strength as unit resolution [Chang, 1970], and because unit propagation is used by CDCL search algorithms to detect conflicts. In fact, the next theorem gives even more. The assumption is that CDCL searches may use any of the clause learning algorithms described in [Marques-Silva and Sakallah, 1999] or [Beame *et al.*, 2004]:

Theorem 5 [Buss *et al.*, 2008] *The regWRTI refutation system and non-greedy CDCL proof search without restarts are polynomially equivalent. Namely, a CDCL search without restarts that finds* Γ *to be unsatisfiable yields a polynomial size regWRTI refutation; and conversely, a regWRTI refutation can be simulated by a non-greedy CDCL search without restarts with the appropriate choices for decision branches and for clause learning.*

Finally, we state the definition of the "guarded graph tautologies" from [Alekhnovich *et al.*, 2007]. These tautologies are based on the graph tautologies GT_n introduced originally by [Krishnamurthy, 1985]. Let $n > 1$ and $[n] = \{0, 1, \ldots, n-1\}$. Let $x_{i,j}$ be variables for distinct $i, j \in [n]$. We identify the negation $\overline{x}_{i,j}$ of $x_{i,j}$ with the variable $x_{j,i}$. Thus there are $\binom{n}{2}$ distinct variables. Let $r(i,j,k)$ and $s(i,j,k)$ be arbitrary functions mapping $[n]^3$ to $[n]$ such that $\{r(i,j,k), s(i,j,k)\} \not\subset \{i,j,k\}$, for all distinct values i, j, k.

Definition 6 GGT_n *is the following (unsatisfiable) set of clauses:*

a. $\bigvee_{j \neq i} x_{j,i}$, *for each value* $i \in [n]$.

b. $\overline{x}_{i,j} \vee \overline{x}_{j,k} \vee \overline{x}_{k,i} \vee x_{r,s}$ *and* $\overline{x}_{i,j} \vee \overline{x}_{j,k} \vee \overline{x}_{k,i} \vee \overline{x}_{r,s}$ *for all distinct* $i, j, k \in [n]$ *and* $r = r(i,j,k)$ *and* $s = s(i,j,k)$.

The intuition behind the unsatisfiability of the GGT_n clauses is that any transitive strict partial order $\prec$ on n vertices must have a minimal element. The intended meaning of $x_{i,j}$ is that it expresses the condition that $i \prec j$. The clauses of type a. express that there is no $\prec$-minimal element i. The clauses of type b. allow the derivation of the *transitive closure* clauses $\overline{x}_{i,j} \vee \overline{x}_{j,k} \vee \overline{x}_{k,i}$ expressing the transitivity of $\prec$. These, plus convention that $x_{i,j}$ is the complement of $x_{j,i}$, implies that $\prec$ is a strict partial order.

The (unguarded) graph tautologies GT_n are defined as the set of clauses of type a. plus the transitivity clauses $\overline{x}_{i,j} \vee \overline{x}_{j,k} \vee \overline{x}_{k,i}$. [Stålmarck, 1996; Bonet and Galesi, 2001; Van Gelder, 2006] proved that the clauses GT_n have polynomial size resolution refutations. For GGT_n, we have:

Theorem 7 [Alekhnovich *et al.*, 2007] *The clauses* GGT_n *have polynomial size resolution refutations, but require size* $\geq 2^{n/10}$ *for regular resolution refutations.*

The fact that the transitive closure clauses must be derived using resolution on $x_{r,s}$ is what makes it impossible to give polynomial size regular refutations of the GGT_n clauses.

4 Statements of Theorems

We can now state our main theorems. For the pool resolution refutation system, we have the following:

Theorem 8 *The guarded graph tautology principles* GGT_n *have polynomial size pool resolution refutations with input lemmas.*

Or equivalently:

Theorem 9 *The guarded graph tautology principles* GGT_n *have polynomial size regular tree-like resolution refutations with input lemmas, i.e. regRTI refutations.*

The refutations of Theorems 8 and 9 do not use degenerate or w-resolution inferences, just ordinary resolution. They are a special case of regWRTI refutations. Thus this theorem is slightly stronger than is actually needed as, in view of Theorem 5, it would be sufficient to have Theorem 9 hold with pool w-resolution refutations instead of pool resolution refutations.

For CDCL proof search, we have:

Theorem 10 *A greedy CDCL proof search, with the correct choices for decision variables, learned clauses and garbage collected clauses, but without restarts, can refute the* GGT_n *clauses in polynomial time.*

It follows that the guarded graph tautologies do not superpolynomially separate CDCL without restarts from resolution.

The proof of the last theorem is via a direct translation from the regRTI refutations of Theorem 9 to CDCL searches.

We conclude with speculations about what these theorems, along with those of [Bonet *et al.*, 2012] and [Buss and Kołodziejczyk, 2012], say about whether CDCL proof search without restarts can polynomially simulate resolution. On one hand, CDCL proof search without restarts is unexpectedly powerful and practical even for large-scale applications, for instance in verification. This gives some hope that perhaps CDCL without restarts is powerful enough to polynomially simulate resolution. In fact, one might speculate that the ability to simulate resolution is one of the reasons for the success of CDCL procedures. A skeptic would counter that the surprising power of CDCL is perhaps due more to its ability to use conflicts to learn clauses intelligently, not any ability to simulate resolution.

On the other hand, all attempts to prove that CDCL without restarts can polynomially simulate resolution have failed so far, and there is no indication that such a polynomial simulation exists. What is needed, however, are examples of tautologies that might plausibly separate CDCL without restarts from resolution.

Acknowledgement. We thank J. Johannsen, L. Kołodziejczyk, and anonymous referees for helpful insights, corrections, and other comments.

References

[Alekhnovich *et al.*, 2007] Michael Alekhnovich, Jan Johannsen, Toniann Pitassi, and Alasdair Urquhart. An exponential separation between regular and general resolution. *Theory of Computing*, 3(5):81–102, 2007.

[Atserias *et al.*, 2011] Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-learning algorithms with many restarts and bounded-width resolution. *Journal of Artificial Intelligence Research*, 40:353–373, 2011.

[Beame *et al.*, 2004] Paul Beame, Henry A. Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *J. Artificial Intelligence Research*, 22:319–351, 2004.

[Bonet and Buss, 2012a] Maria Luisa Bonet and Samuel R. Buss. An improved separation of regular resolution from pool resolution and clause learning. In *Proc. 15th International Conference on Theory and Applications of Satisfiability Testing – SAT 2012*, Lecture Notes in Computer Science #7317, pages 45–57, 2012.

[Bonet and Buss, 2012b] Maria Luisa Bonet and Samuel R. Buss. An improved separation of regular resolution from pool resolution and clause learning. Full version, arxiv.org, arXiv:1202.2296v2 [cs.LO], 2012.

[Bonet and Galesi, 2001] Maria Luisa Bonet and Nicola Galesi. Optimality of size-width tradeoffs for resolution. *Computational Complexity*, 10(4):461–474, 2001.

[Bonet *et al.*, 2012] Maria Luisa Bonet, Samuel R. Buss, and Jan Johannsen. Improved separations of regular resolution from pool resolution and clause learning. Typeset manuscript, 2012.

[Buss and Kołodziejczyk, 2012] Sam Buss and Leszek Kołodziejczyk. Small stone in pool. Preprint, 2012.

[Buss *et al.*, 2008] Samuel R. Buss, Jan Hoffmann, and Jan Johannsen. Resolution trees with lemmas: Resolution refinements that characterize DLL-algorithms with clause learning. *Logical Methods in Computer Science*, 4, 4:13(4:13):1–18, 2008.

[Chang, 1970] C. L. Chang. The unit proof and the input proof in theorem proving. *J. ACM*, 17(4):698–707, 1970.

[Davis and Putnam, 1960] Martin Davis and Hillary Putnam. A computing procedure for quantification theory. *Journal of the Association for Computing Machinery*, 7(3):201–215, 1960.

[Davis *et al.*, 1962] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, 1962.

[Gomes *et al.*, 2000] Carla P. Gomes, Bart Selman, Nuno Crato, and Henry Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of Automated Reasoning*, 24:67–100, 2000.

[Hertel *et al.*, 2008] Philipp Hertel, Fahim Bacchus, Toniann Pitassi, and Allen Van Gelder. Clause learning can effectively p-simulate general propositional resolution. In *Proc. 23rd AAAI Conf. on Artificial Intelligence (AAAI 2008)*, pages 283–290. AAAI Press, 2008.

[Krishnamurthy, 1985] Balakrishnan Krishnamurthy. Short proofs for tricky formulas. *Acta Informatica*, 22(3):253–275, 1985.

[Marques-Silva and Sakallah, 1999] João P. Marques-Silva and Karem A. Sakallah. GRASP — A new search algorithm for satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, 1999.

[Pipatsrisawat and Darwiche, 2011] Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, 172(2):512–525, 2011.

[Stålmarck, 1996] Gunnar Stålmarck. Short resolution proofs for a sequence of tricky formulas. *Acta Informatica*, 33(3):277–280, 1996.

[Urquhart, 2011] Alasdair Urquhart. A near-optimal separation of regular and general resolution. *SIAM Journal on Computing*, 40(1):107–121, 2011.

[Van Gelder, 2005] Allen Van Gelder. Pool resolution and its relation to regular resolution and DPLL with clause learning. In *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2005)*, Lecture Notes in Computer Science 3835, pages 580–594. Springer-Verlag, 2005.

[Van Gelder, 2006] Allen Van Gelder. Preliminary report on input cover number as a metric for propositional resolution proofs. In *Theory and Applications of Satisfiability Testing - SAT 2006*, Lecture Notes in Computer Science 4121, pages 48–53. Springer Verlag, 2006.